# Equatorial Platform II

Design Group SD0921

An equatorial platform is used to allow an amateur astronomer to track a celestial body. Typical mounts commercially available are latitude specific. A previous design group designed a simple platform that is not latitude specific and we propose describe and improve on their design.

**Kayla Helseth, Mark Jund, Justin Sipma, Nathan McBeth**

**5/7/2010**

# Table of Contents

## Introduction:

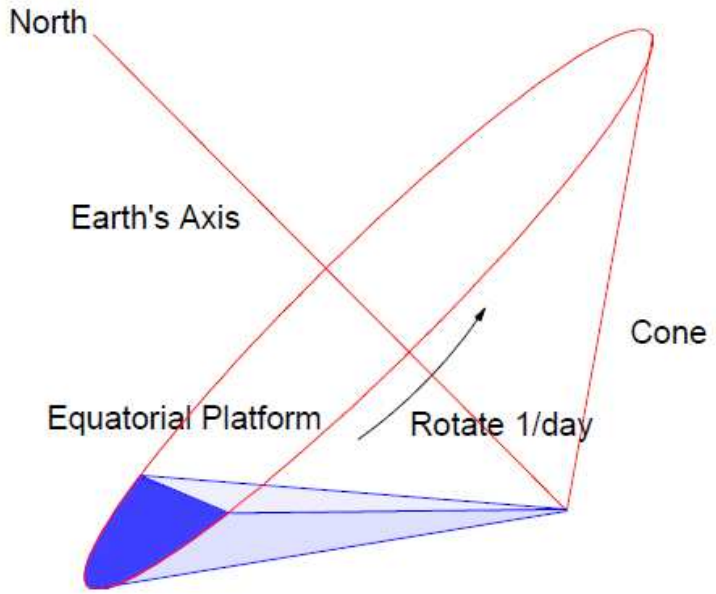
Celestial bodies fill the sky at night. A multitude of stars are visible, as well as a few close planets, the moon and the sun in the day. A telescope can be used by amateur astronomers to view these bodies more closely. However, as the viewer looks through the eyepiece, the earth rotates and the star or body in question quickly moves out of the eyepiece.

The earth completes one rotation in 24 hours or 1440 minutes. The sun, moon and stars each move at a slightly different rate due to their varying distance from earth and what they rotate about. In order for an astronomer to view a body for an extended period of time, they must continue to move the telescope to follow the earth's rotation.

An equatorial platform solves this problem by automatically moving the telescope at the same rate the body is moving across the sky. The speed is latitude dependant and our platform can be altered to function at any latitude.

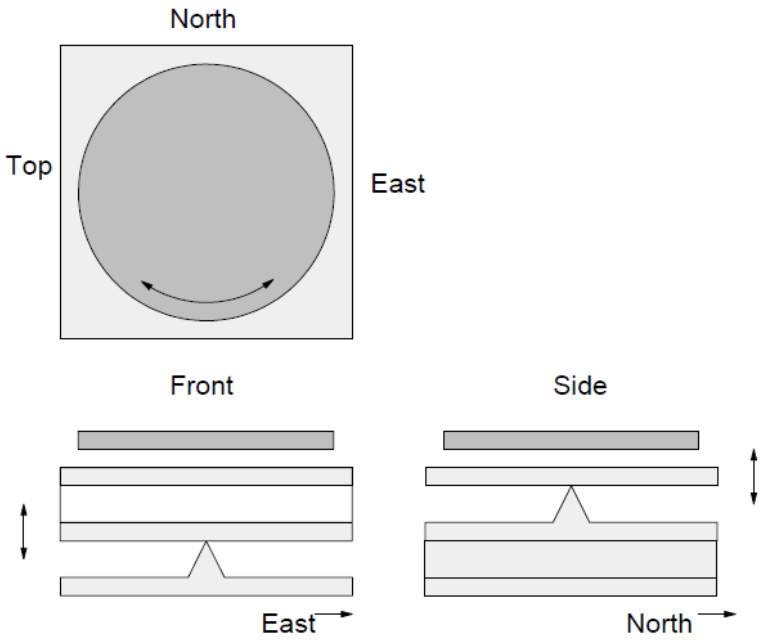
Our purpose was to design a platform that could be used at any latitude and operate for longer than twenty minutes. This platform could be used by amateur astronomers to allow for prolonged viewing of the sun, moon or stars. It will not be technically challenging to operate so a user could operate it without any prior knowledge of its design.

One way to design such a platform is to take a cone pointed at true north as presented in the following Figure. If you take a section of that cone, you have a platform to place telescopes, cameras, or whatever you like. By rotating that section as if it were part of the larger cone, stars will appear motionless.



## Theory:

In principle, the conic section defined by Georges d'Autume defines the plane the equatorial platform is to implement. This can be done with a conic. It can also be implemented with two planes and a pivot:




By tilting the platform on an east-west axis as well as a north-south axis, any plane can be obtained. Motors attached to each axis tilt each side to match the normal to the plane defined by the conic-section equatorial platform. In addition to defining a plane, the conic-section platform rotates the plane. To accomplish this, a third axis is added: the top of the platform can rotate, providing this motion.

The advantage of this proposed design is the simplicity in construction: the previous group used two hinged joints along with a rotational plane. We simplified it further by using a single plane that can tilt east-west and north-south along with a rotational plane. As a result, the need for a complex conic section is avoided. Second, such a platform will work at any latitude. All that needs to be changed is the software to drive the motors if one changes locations. The disadvantage is that three motors are required along with a more complex algorithm to determine the position of each axis at any time.

## Requirements

We were asked to improve on the past group's design. Our platform needed to be large enough to accommodate a 150 pound, 18" Dobsonian mount telescope, limit power consumption, have a cleaner appearance and track the sun, moon and stars as opposed to just the sun. The past design functioned relatively well, but we were asked to create a more user friendly and versatile platform.

# Design

## Equator

If you are at the equator, the stars rotate from east to west. There is no north/south motion and there is no rotation. As a result, the motion of the platform to track the sun for one hour is

NS: no motion
EW: -7.5 degrees to +7.5 degrees
Z: no rotation.

To track the stars, the motion is:

NS: no motion
EW: -7.520556 to +7.520556
Z: no rotation

To track the moon, the motion is:

NS: no motion
EW: -7.22 degrees to 7.222 degrees
Z: no rotation

## North Pole

If you are at the North Pole, the stars rotate about the zenith. There is no up/down motion whatsoever.

As a result, to track the sun for one hour, the platform should move as:

NS: no motion
EW: no rotation.
Z: -7.5 degrees to +7.5 degrees

To track the stars, the motion is:

NS: no motion
EW: no rotation
Z: -7.520556 to +7.520556

To track the moon, the motion is:

NS: no motion
EW: no rotation
Z: -7.22 degrees to 7.222 degrees

## Hypothesis

Since the Earth is a sphere, the motion for latitude, Q, should be:

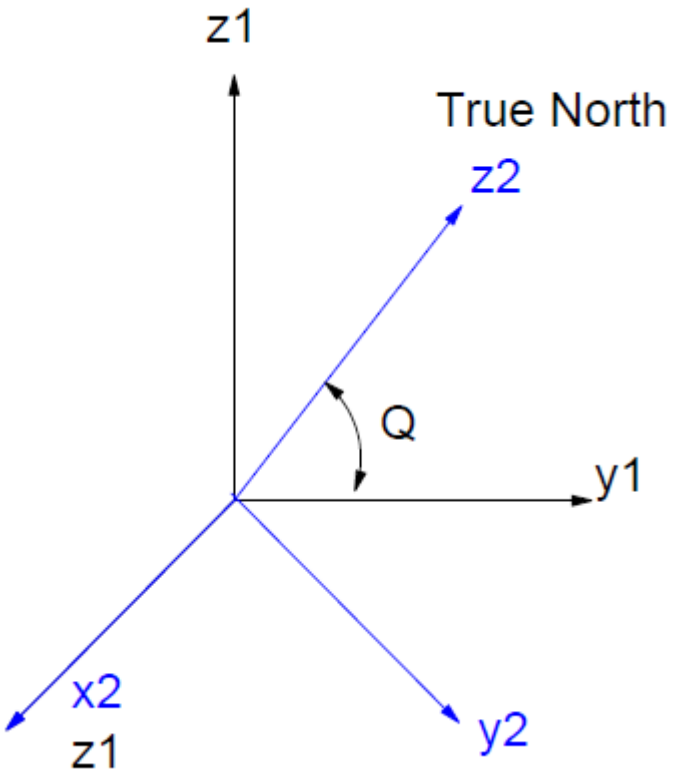
NS: no motion
EW: θ*cos(*Q*)
Z: θ*sin(*Q*)

Where θ goes from the beginning of each range to the end for each celestial body.

## Computations:

To verify this, consider the equations of motion for a conic-section equatorial platform at latitude Q.

Align axis (x1, y1, z1) so that the z1 axis aligns with true north. For Fargo, ND, Q = 46.89 degrees.

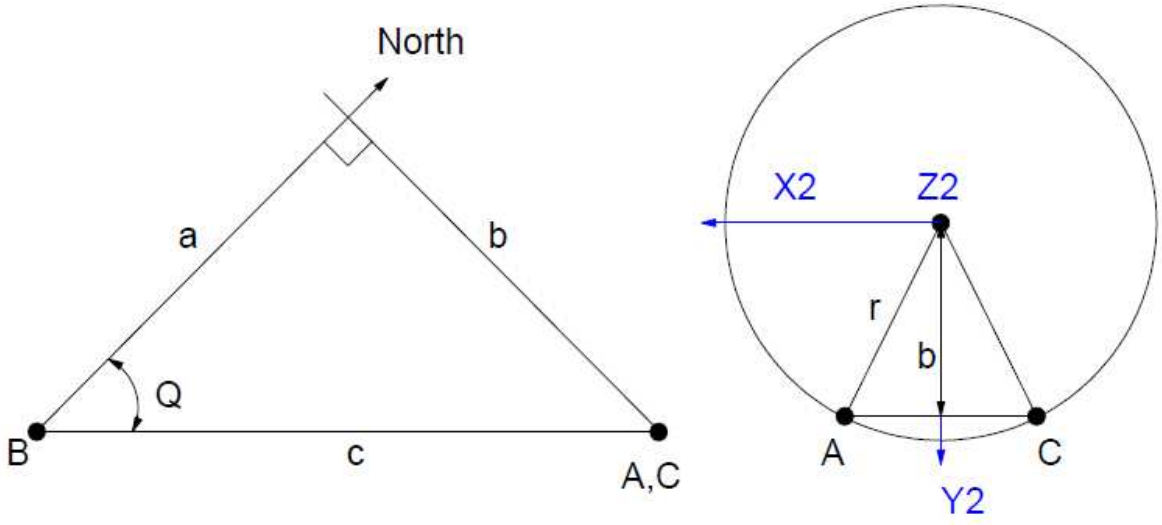



The transformation from coordinate system 1 to 2 is:

$$\begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} = T_{21} \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_1) & \sin(\theta_1) \\ 0 & -\sin(\theta_1) & \cos(\theta_1) \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix}$$

where θ1 = 90º – *Q*.

The size of the platform doesn't actually matter. All that matters is the normal to the platform. This normal will be duplicated with the platform we're designing. Assume a platform which is 1 meter from the tip of the conic section:




Then

*a* = *c* cos(*Q*)

*b* = *c* sin(*Q*)

If ABC=60 degrees and c=1,

$$r^2 = b^2 + \left(\frac{r}{2}\right)^2$$

$$r^2 = \frac{4}{3}b^2$$

$$\varphi = \cos^{-1}\left(\frac{b}{r}\right) = 30^0$$

Point A moves about this circle for 1 hour, so in coordinate system 1:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}_{A2} = \begin{pmatrix} r \cdot \sin(\phi + \varphi) \\ r \cdot \cos(\phi + \varphi) \\ a \end{pmatrix}$$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}_{C2} = \begin{pmatrix} r \cdot \sin(\phi - \varphi) \\ r \cdot \cos(\phi - \varphi) \\ b \end{pmatrix}$$

where $-7.50 < \phi < 7.50$ to rotate one hour (for the sun).

In Earth coordinates:

Taking the normal to the plane:

$$Z_1 = A_1 \, x \, C_1 = \begin{vmatrix} x & y & z \\ A_x & A_y & A_z \\ C_x & C_y & C_z \end{vmatrix}$$

When normalized, the component in the x direction tells you the angle that the platform should tilt in the E/W direction. The component in the y direction tells you the angle that the platform should tilt in the N/S direction.

## Equator

Example: Assume you are almost at the equator

Q = 1 degree r= 0.0202
a = 0.0175 b= 0.9998
c = 1.0000
Z1 =

| | | |
|---|---|---|
| - 0.1305063 | 0.0001493 | 0.9914475 |
| - 0.0871425 | 0.0000664 | 0.9961959 |
| - 0.0436127 | 0.0000166 | 0.9990485 |
| 0. | - 8.674D-17 | 1. |
| 0.0436127 | 0.0000166 | 0.9990485 |
| 0.0871425 | 0.0000664 | 0.9961959 |
| 0.1305063 | 0.0001493 | 0.9914475 |

## North Pole

Example: Assume you are almost at the North pole:

Q = 89 degrees a= 0.0174524
b = 0.9998477 r= 1.1545247
Z1 =

| | | |
|---|---|---|
| - 0.0022780 | 0.0001493 | 0.9999974 |
| - 0.0015211 | 0.0000664 | 0.9999988 |
| - 0.0007613 | 0.0000166 | 0.9999997 |
| 0. | - 6.245D-17 | 1. |
| 0.0007613 | 0.0000166 | 0.9999997 |
| 0.0015211 | 0.0000664 | 0.9999988 |
| 0.0022780 | 0.0001493 | 0.9999974 |

## Fargo

Example: Assume you are in Fargo, ND (Q = 46.89 degrees)

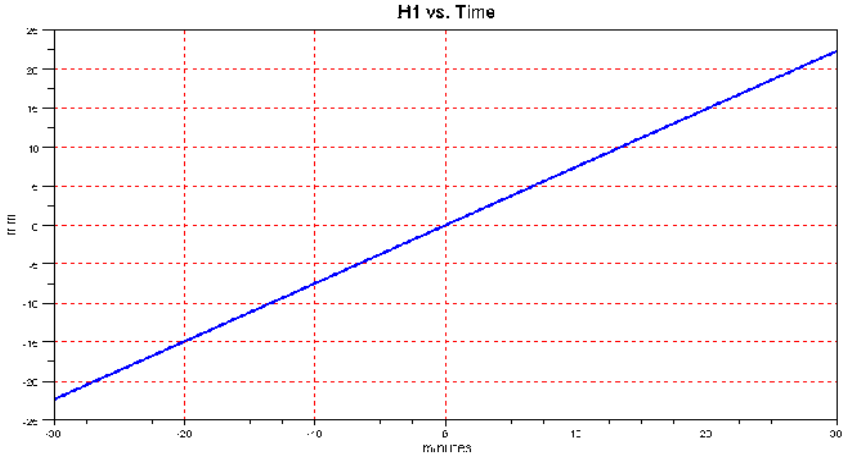
Q = 46.89 degrees r= 0.8429811
a = 0.6834012 b= 0.7300430
c = 1.0000000
Z1=

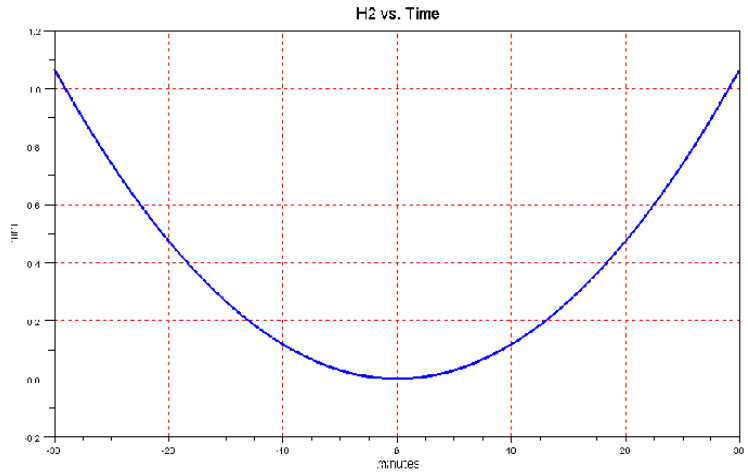
| | | |
|---|---|---|
| - 0.0892018 | 0.0042683 | 0.9960044 |
| - 0.0595623 | 0.0018985 | 0.9982228 |
| - 0.0298095 | 0.0004749 | 0.9995555 |
| 0. | - 1.665D-16 | 1. |
| 0.0298095 | 0.0004749 | 0.9995555 |
| 0.0595623 | 0.0018985 | 0.9982228 |
| 0.0892018 | 0.0042683 | 0.9960044 |

## Example: Fargo, ND

NS / EW / Z axis motion for one hour tracking

| Sun | Moon | Stars |
|---|---|---|
| EW: | EW: | EW: |
| • 8.154 cm/hour<br>• 10274 steps/hour<br>• 2.854 steps/second | • 8.154 cm/hour<br>• 10274 steps/hour<br>• 2.854 steps/second | • 8.154 cm/hour<br>• 10274 steps/hour<br>• 2.854 steps/second |
| NS: | NS: | NS: |
| • $0.27297(T^2)$ cm<br>• -20 min <T< 20 min | • $0.266948(T^2)$ cm<br>• -20 min <T< 20 min | • $0.27297(T^2)$ cm<br>• -20 min <T< 20 min |
| Z | Z | Z |
| • 2.4 cm/hour<br>• 76.3938 steps/hour<br>• .0212 steps/second | • 2.3124 cm/hour<br>• 73.606 steps/hour<br>• .0204 steps/second | • 2.406 cm/hour<br>• 76.5853 steps/hour<br>• .0213 steps/second |






## Detailed Motion Example for each axis for the Sun:

### East – West Board:

| | EW Tilt (X): Degrees ( = 7.4785885 * cos(Q)) | | | | | | |
|---|---|---|---|---|---|---|---|
| minutes | 0.1 | 15 | 30 | 45 | 60 | 75 | 89.9 |
| -30 | -7.479 | -7.224 | -6.477 | -5.288 | -3.739 | -1.936 | 0.000 |
| -20 | -4.994 | -4.824 | -4.325 | -3.531 | -2.497 | -1.292 | 0.000 |
| -10 | -2.499 | -2.414 | -2.164 | -1.767 | -1.250 | -0.647 | 0.000 |
| 0 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 10 | 2.499 | 2.414 | 2.164 | 1.767 | 1.250 | 0.647 | 0.000 |
| 20 | 4.994 | 4.824 | 4.325 | 3.531 | 2.497 | 1.292 | 0.000 |
| 30 | 7.479 | 7.224 | 6.477 | 5.288 | 3.739 | 1.936 | 0.000 |

## North South Board:

| | NS Tilt (Y): Degrees = 0.4902 * sin(Q)cos(Q) | | | | | | |
|---|---|---|---|---|---|---|---|
| minutes | 0.1 | 15 | 30 | 45 | 60 | 75 | 89.9 |
| -30 | 0.000 | 0.12254 | 0.212 | 0.245 | 0.212 | 0.123 | 0.000 |
| -20 | 0.000 | 0.055 | 0.094 | 0.109 | 0.094 | 0.055 | 0.000 |
| -10 | 0.000 | 0.014 | 0.024 | 0.027 | 0.024 | 0.014 | 0.000 |
| 0 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 10 | 0.000 | 0.014 | 0.024 | 0.027 | 0.024 | 0.014 | 0.000 |
| 20 | 0.000 | 0.055 | 0.094 | 0.109 | 0.094 | 0.055 | 0.000 |
| 30 | 0.000 | 0.123 | 0.212 | 0.245 | 0.212 | 0.123 | 0.000 |

## The Rotational Board:

| | Z: Rotation: Origin to B1 (degrees) | | | | | | |
|---|---|---|---|---|---|---|---|
| minutes | 0.1 | 15 | 30 | 45 | 60 | 75 | 89.9 |
| -30 | 0.013 | 1.936 | 3.739 | 5.288 | 6.477 | 7.224 | 7.479 |
| -20 | 0.009 | 1.292 | 2.497 | 3.531 | 4.325 | 4.824 | 4.994 |
| -10 | 0.004 | 0.647 | 1.250 | 1.767 | 2.164 | 2.414 | 2.499 |
| 0 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 10 | -0.004 | -0.647 | -1.250 | -1.767 | -2.164 | -2.414 | -2.499 |
| 20 | -0.009 | -1.292 | -2.497 | -3.531 | -4.325 | -4.824 | -4.994 |
| 30 | -0.013 | -1.936 | -3.739 | -5.288 | -6.477 | -7.224 | -7.479 |

Note that these results fit the following equations (where Q is your latitude and t is the time in minutes, ranging from -30 to +30):

$$EW = 7.4785885 \cdot \cos(Q) \cdot \left(\frac{t}{30}\right)$$

$$NS = 0.4902 * \sin(Q)\cos(Q) * \left(\frac{t}{30}\right)^2$$

$$Z = -7.47859 \cdot \sin(Q) \cdot \left(\frac{t}{30}\right)$$

As a result, the proposed equatorial platform will operate at any latitude.

The results for the stars is

$$EW = 7.520556 \cos Q * \left(\frac{t}{30}\right)$$

$$NS = 0.501 \sin Q \cos Q \left(\frac{t}{30}\right)^2$$

$$Z = -7.520556 \sin Q \left(\frac{t}{30}\right)$$

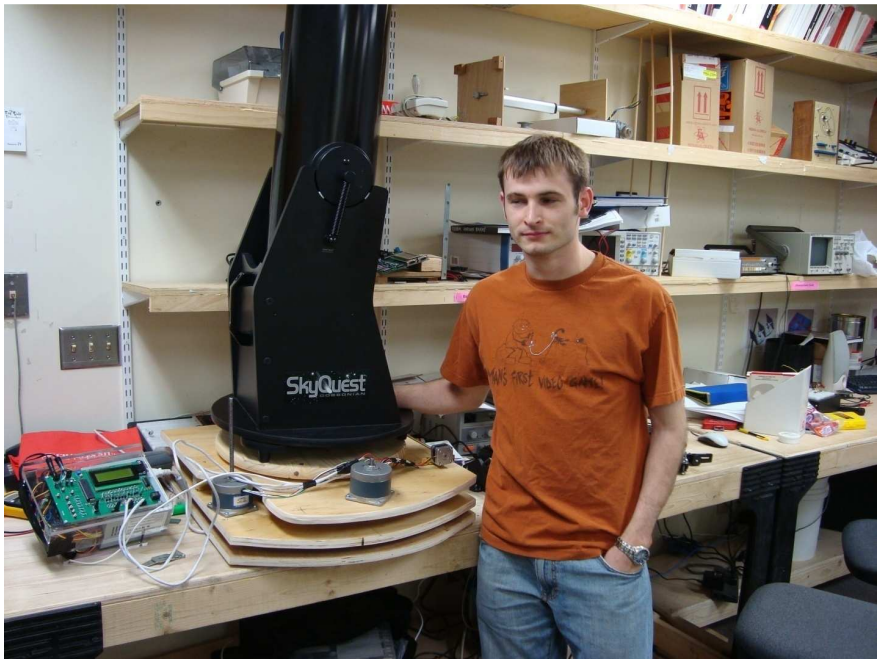
And for the moon we have

$$EW = 7.222 \cos Q * \left(\frac{t}{30}\right)$$

$$NS = 0.48 \sin Q \cos Q \left(\frac{t}{30}\right)^2$$

$$Z = -7.222 \sin Q \left(\frac{t}{30}\right)$$

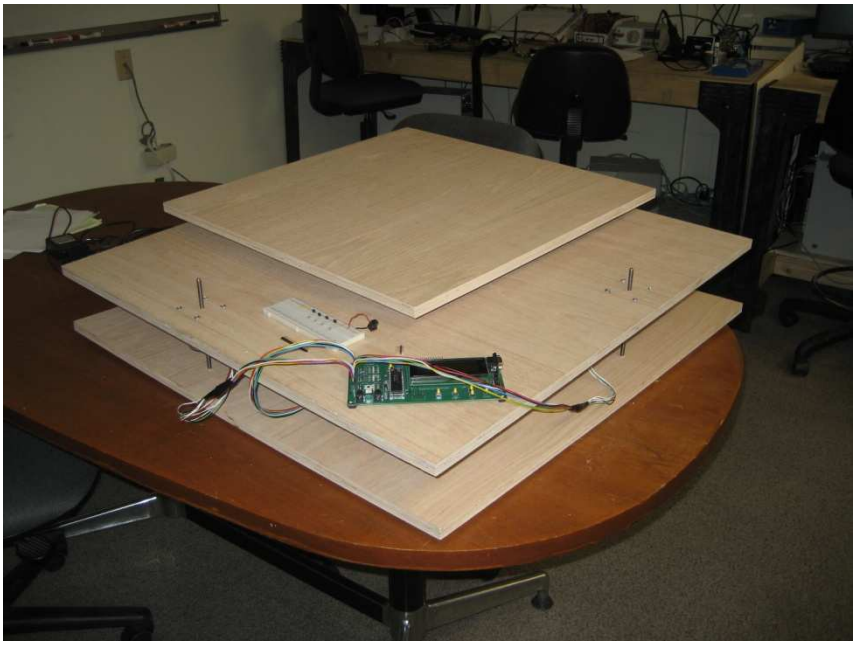
## Initial Platform Prototype




## Specifications:

- Weight: 50 lb
- Size: 22” x 22” x 8”
- Latitude: 0 to 90 degrees

## Secondary Platform Prototype:



## Specifications:

- Weight: 60 lbs
- Size: 33” x 33” x 8”
- Latitude: 0 to 90 degrees

## Application:

Assume that each arm for the equatorial platform is 250mm. Then

| | EW Tilt (X): millimeters | | | | | | |
|---|---|---|---|---|---|---|---|
| minutes | 0.1 | 15 | 30 | 45 | 60 | 75 | 89.9 |
| -30 | -51.199 | -49.434 | -44.276 | -36.097 | -25.487 | -13.183 | 0.000 |
| -20 | -34.079 | -32.914 | -29.495 | -24.065 | -17.007 | -8.796 | 0.000 |
| -10 | -17.021 | -16.441 | -14.737 | -12.031 | -8.510 | -4.404 | 0.000 |
| 0 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 10 | 17.021 | 16.441 | 14.737 | 12.031 | 8.510 | 4.404 | 0.000 |
| 20 | 34.079 | 32.914 | 29.495 | 24.065 | 17.007 | 8.796 | 0.000 |
| 30 | 51.199 | 49.434 | 44.276 | 36.097 | 25.487 | 13.183 | 0.000 |

| | NS Tilt (Y): millimeters | | | | | | |
|---|---|---|---|---|---|---|---|
| minutes | 0.1 | 15 | 30 | 45 | 60 | 75 | 89.9 |
| -30 | 0.000 | 0.834 | 1.443 | 1.668 | 1.443 | 0.837 | 0.000 |
| -20 | 0.000 | 0.374 | 0.640 | 0.742 | 0.640 | 0.374 | 0.000 |
| -10 | 0.000 | 0.095 | 0.163 | 0.184 | 0.163 | 0.095 | 0.000 |
| 0 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 10 | 0.000 | 0.095 | 0.163 | 0.184 | 0.163 | 0.095 | 0.000 |
| 20 | 0.000 | 0.374 | 0.640 | 0.742 | 0.640 | 0.374 | 0.000 |
| 30 | 0.000 | 0.837 | 1.443 | 1.668 | 1.443 | 0.837 | 0.000 |

| | Z: Rotation: millimeters (280mm lever) | | | | | | |
|---|---|---|---|---|---|---|---|
| minutes | 0.1 | 15 | 30 | 45 | 60 | 75 | 89.9 |
| -30 | 0.064 | 9.465 | 18.298 | 25.916 | 31.788 | 35.491 | 36.758 |
| -20 | 0.044 | 6.315 | 12.210 | 17.278 | 21.176 | 23.630 | 24.467 |
| -10 | 0.020 | 3.162 | 6.110 | 8.638 | 10.580 | 11.804 | 12.220 |
| 0 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 10 | -0.020 | -3.162 | -6.110 | -8.638 | -10.580 | -11.804 | -12.220 |
| 20 | -0.044 | -6.315 | -12.210 | -17.278 | -21.176 | -23.630 | -24.467 |
| 30 | -0.064 | -9.465 | -18.298 | -25.916 | -31.788 | -35.491 | -36.758 |

Assuming a 1/4 x 20 screw and a stepper motor with 400 steps / rotation:

| | EW Tilt (X): steps vs. Latitude | | | | | | |
|---|---|---|---|---|---|---|---|
| minutes | 0.1 | 15 | 30 | 45 | 60 | 75 | 89.9 |
| -30 | -16,126 | -15,569 | -13,946 | -11,370 | -8,028 | -4,151 | 0 |
| -20 | -10,734 | -10,365 | -9,291 | -7,581 | -5,357 | -2,772 | 0 |
| -10 | -5,361 | -5,178 | -4,643 | -3,789 | -2,680 | -1,386 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 5,361 | 5,178 | 4,643 | 3,789 | 2,680 | 1,386 | 0 |
| 20 | 10,734 | 10,365 | 9,291 | 7,581 | 5,357 | 2,772 | 0 |
| 30 | 16,126 | 15,569 | 13,946 | 11,370 | 8,028 | 4,151 | 0 |

| | NS Tilt (Y): steps vs. Latitude | | | | | | |
|---|---|---|---|---|---|---|---|
| minutes | 0.1 | 15 | 30 | 45 | 60 | 75 | 89.9 |
| -30 | 0 | 263 | 454 | 525 | 454 | 264 | 0 |
| -20 | 0 | 118 | 202 | 234 | 202 | 118 | 0 |
| -10 | 0 | 30 | 51 | 58 | 51 | 30 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 0 | 30 | 51 | 58 | 51 | 30 | 0 |
| 20 | 0 | 118 | 202 | 234 | 202 | 118 | 0 |
| 30 | 0 | 264 | 454 | 525 | 454 | 264 | 0 |

| | Z: Rotation: steps vs. Latitude | | | | | | |
|---|---|---|---|---|---|---|---|
| minutes | 0.1 | 15 | 30 | 45 | 60 | 75 | 89.9 |
| -30 | 20 | 2,981 | 5,763 | 8,163 | 10,012 | 11,178 | 11,577 |
| -20 | 14 | 1,989 | 3,846 | 5,442 | 6,670 | 7,443 | 7,706 |
| -10 | 6 | 996 | 1,924 | 2,721 | 3,332 | 3,718 | 3,849 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | -6 | -996 | -1,924 | -2,721 | -3,332 | -3,718 | -3,849 |
| 20 | -14 | -1,989 | -3,846 | -5,442 | -6,670 | -7,443 | -7,706 |
| 30 | -20 | -2,981 | -5,763 | -8,163 | -10,012 | -11,178 | -11,577 |

Our platform has 45.72 cm moment arm. The E/W and N/S motors have 200 steps/rotation and the Z rotation motor has 400 steps/rotation with a 10.7 gear ratio. These were figured into our calculations.

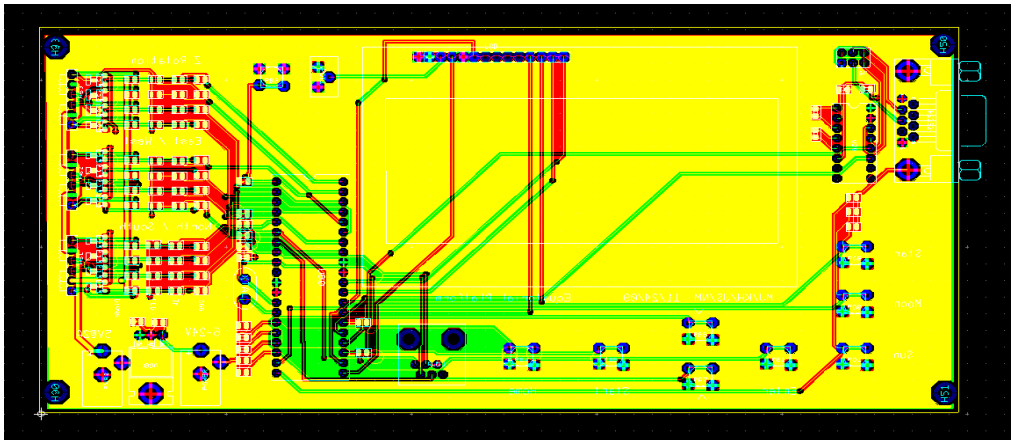
## Hardware:

- Three Linear Actuators
- PIC microcontroller
- Plywood
- One Shaft Coupler
- One Lazy Susan
- Assorted Nuts and Bolts
- Two Universal Joints
- One level
- One compass

## Electronics

- PIC18F4620 microcontroller running at 20MHz
- Transistors driving the stepper motors in unipolar configuration
- LCD display
- Push buttons for reset, and sun, moon or stars.
- PCB layout in Ultiboard

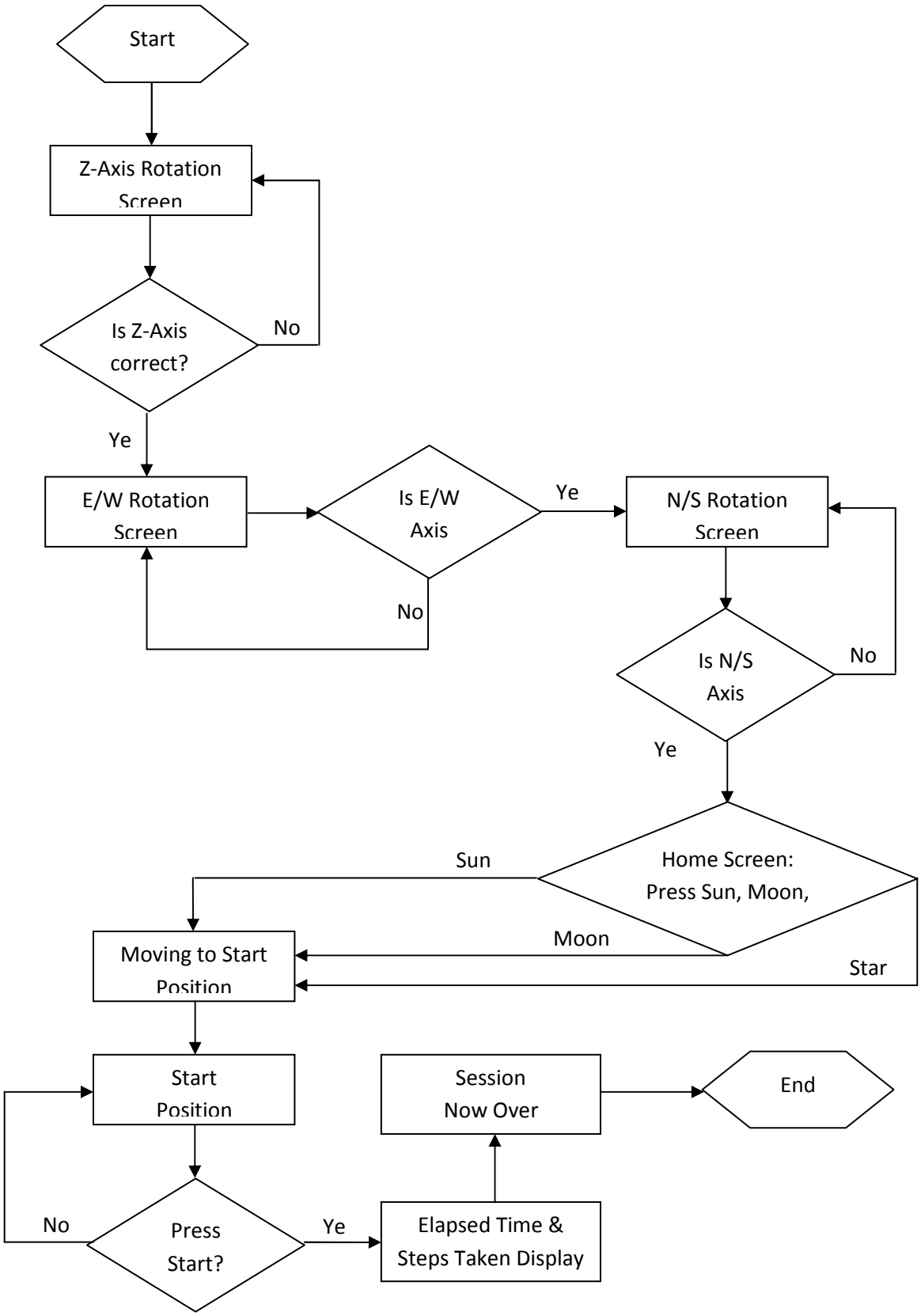
## Layout



## Software

- Half stepping used for stepper motors
- Timer2 interrupts implemented
- LCD display
    - Keeps track of steps taken
    - Allows for the orientation of the board to become level
- Programmed to run for 40 minutes and our platform can support that without binding.
- Software Alterations
    - Added different step sizes for different celestial bodies since the sun, moon and stars all move at slightly different speeds.
    - Adjust for different platform size.
    - Adjust for longer run time
- Include programming for new buttons

## Flowchart


Start
Z-Axis Rotation Screen
Is Z-Axis correct?
No
Ye
E/W Rotation Screen
Is E/W Axis
Ye
No
N/S Rotation Screen
Is N/S Axis
No
Ye
Home Screen: Press Sun, Moon,
Sun
Moon
Star
Moving to Start Position
Start Position
Press Start?
No
Ye
Elapsed Time & Steps Taken Display
Session Now Over
End

## Design Features:

- Latitude adjustable
- Track sun, moon or stars for 40 minutes
- Powerful and sturdy to withstand large telescopes
- High degree of accuracy and precision – for smooth images
- Weight & size of the platform – Transportable
- Handle and wheels for ease of movement
- Different from other patented versions
- User Friendly
- Cost effective
- Bulls Eye Level and Compass for easy leveling and positioning

## Troubleshooting:

**Problem**:

* Device does not turn on.

**Solution**:

* Check to make sure the board's power supply is plugged into a working outlet.
* Check to make sure the board's power supply is plugged into the board port.
    - The board must be powered with the 12 volt power supply.
* The contrast on the screen may have been turned all the way down so adjust the dial on the potentiometer.
    - More information regarding potentiometer in "Other Features."

**Problem**:

* The board has power but the motors do not move.

**Solution**:

* Make sure the motor cables are plugged into the socket on the board.
* Make sure the motor's power supply is plugged into a working outlet.
* Make sure the motor's power supply is plugged into the motor port on the board.
    - The motors need to be supplied with the 6 volt power supply.

**Problem**:

* One of the motors operates but it is going in the wrong direction.

**Solution**:

* The motor cable is plugged into the board backwards.

**Problem**:

- The platform fails to move in one direction.

**Solution**:

- The motor is bound up on the screw. To fix, reverse the direction of the motor manually and apply pressure in that direction.

**Problem:**

- The platform says it's moving one direction when really it's moving a different direction.

**Solution:**

- Make sure motor cables are plugged into the proper sockets.
  - They are color-coordinated to help avoid confusion.

**Problem:**

- The board freezes up and none of the buttons respond when being pressed.

**Solution:**

- Press reset and start the program all over.
- Board may have overheated so turn the PIC board off and leave off for at least four hours.

**IMPORTANT**

THE PIC BOARD MUST BE POWERED BY THE 12 VOLT POWER SUPPLY. LIKEWISE, THE MOTORS MUST ONLY USE THE 5 VOLT POWER SUPPLY. MIXING UP THESE TWO CABLES AND PLUGGING THEM INTO THE WRONG PORTS COULD DAMAGE THE BOARD BEYOND REPAIR.

## Project Comments:

### Budget:

| Equatorial Platform | |
|---|---|
| Foamboard | ~$10 |
| Original Hinge | $25 |
| Foamboard | $24 |
| Plywood | $41.79 |
| Shaft Coupler | $30 |
| Bolts, Nuts, etc | $12 |
| 1st Belt | $19 |
| PIC board | $33 |
| 2nd Belt | $35 |
| Lazy Susan | $10 |
| Linear Actuator Bolt Mounts | $45 |
| Compass | $5 |
| Bull's Eye Level | $3 |
| Mobile Hardware | $25 |
| Total | $317.79 |

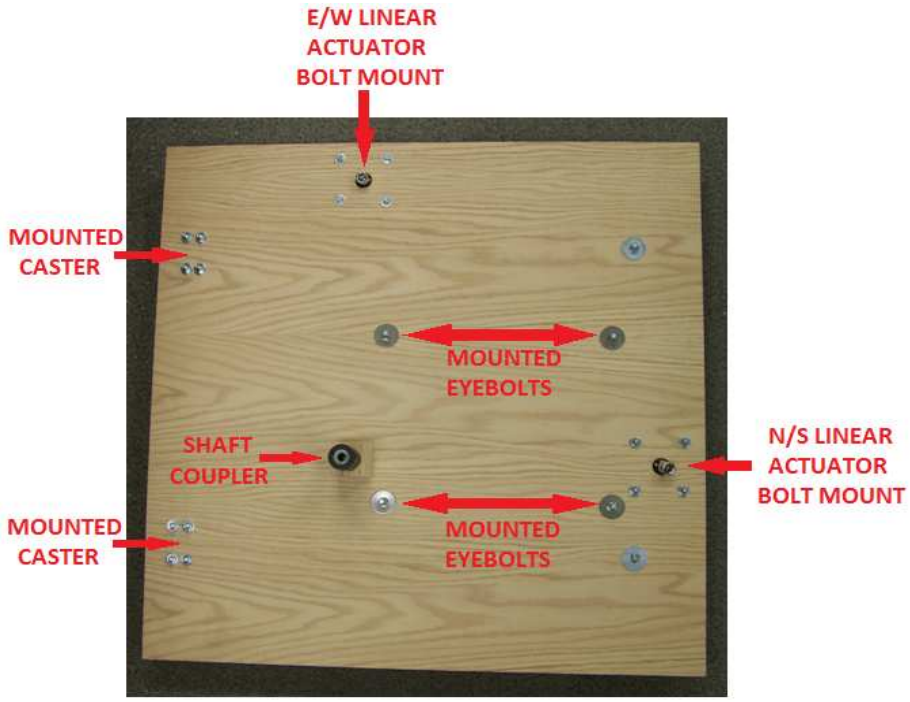
## Testing Results:

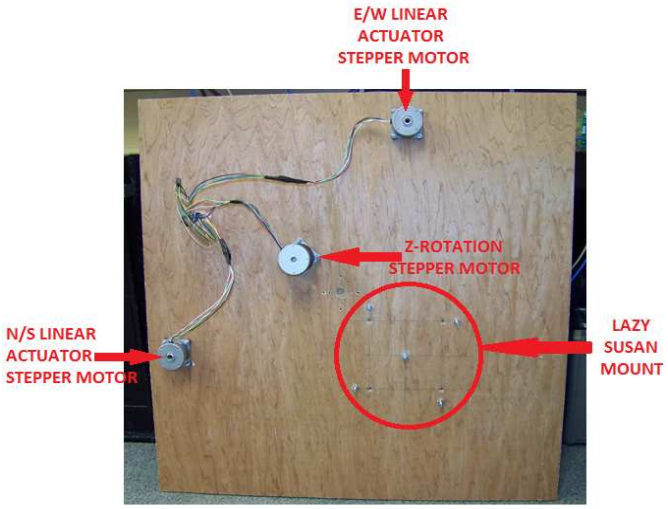
- Power Off
  - Stars, sun and moon quickly drift across the eyepiece's field of view as the earth rotates
- Power On
  - Very slow drift of stars, sun or moon
  - The concept works!
  - The north alignment was probably off and there might have been some rounding error in the calculations

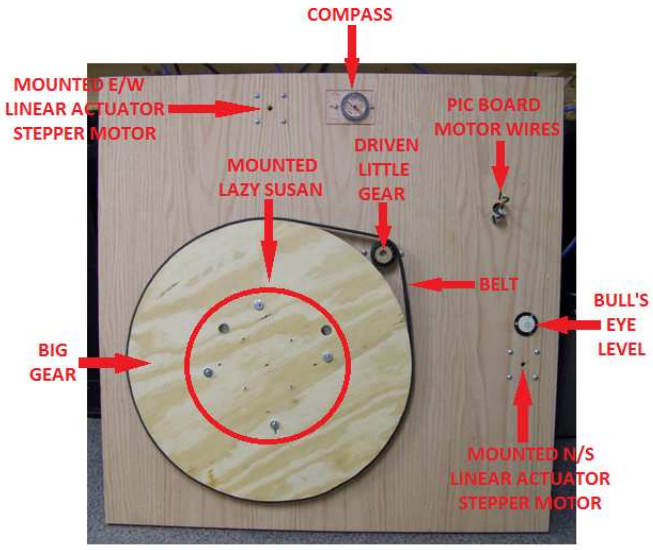
## Future Work

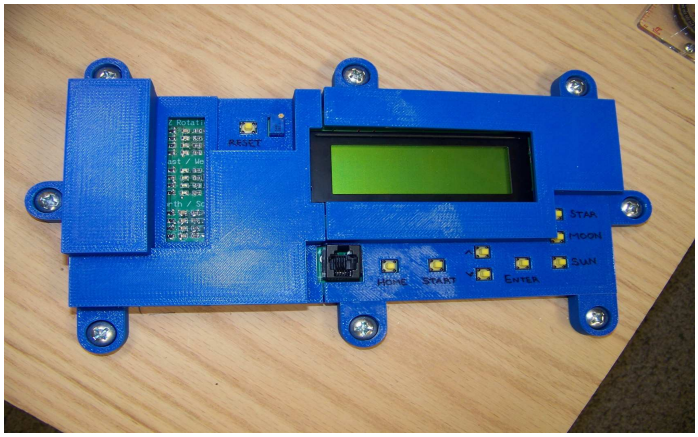
- Hand Controller
- Autoguider
- Allow for user inputted latitude
- Battery or solar operation

## Photos





Whole Enclosure

Pic Board

## Appendix

```
// Equatorial Platform
// Final Draft
//
// Nathan McBeth
// Kayla Helseth
// Mark Jund
// Justin Sipma
//
// 40 minute operation
//
// 46.59 degrees North (Fargo)
//
//       MOON                  STAR                    SUN
// EW = 3,307steps         // EW = 3,444 steps        // EW = 3,424 steps
//    = 1.3126 cm          //    = 1.3669 cm          //    = 1.359 cm
//  Z = 40 steps           //  Z = 42 steps           //  Z = 42 steps
//    = 4.894 cm           //    = 5.090 cm           //    = 5.07 cm
// NS = 107 steps          // NS = 109 steps          // NS = 109 steps
//    = 0.0847 cm          //    = 0.0886 cm          //    = 0.0867 cm
//
// Z rotation based on a 3:31 gear ratio
//
// All steps are per 20 minutes
//
// 10ms step size
//
// T = -120,000 to +120,000

// Global Variables

unsigned char MA0[21] = "      Z Axis      ";
unsigned char MA1[21] = "Use arrows to center";
unsigned char MA2[21] = "  Enter when done   ";
unsigned char MA3[21] = "                    ";

unsigned char MB0[21] = "     E/W Level      ";
unsigned char MB1[21] = "Use arrows to level ";
unsigned char MB2[21] = "  Enter when done   ";
```

```
unsigned char MC0[21] = "     N/S Level      ";
unsigned char MC1[21] = "Use arrows to level ";
unsigned char MC2[21] = "  Enter when done   ";

unsigned char MD0[21] = "Now at Home Position";
unsigned char MD1[21] = "Press Sun, Moon, or ";
unsigned char MD2[21] = "  Star to continue  ";

unsigned char ME0[21] = "  Moving to Start   ";
unsigned char ME1[21] = "      Position      ";

unsigned char MF0[21] = "  Start Position:   ";
unsigned char MF1[21] = "  Press Start to    ";
unsigned char MF2[21] = "  begin tracking    ";

unsigned char MG0[21] = "Time:               ";
unsigned char MG1[21] = " E/W:               ";
unsigned char MG2[21] = " N/S:              ";
unsigned char MG3[21] = "   Z:             ";

unsigned char MH0[21] = "   Moving to Home   ";
unsigned char MH1[21] = "      Position      ";

unsigned char MI0[21] = "Program now complete";
unsigned char MI1[21] = " Press Reset if you ";
unsigned char MI2[21] = " wish to start over ";

unsigned char MJ0[21] = "  Or move back to   ";
unsigned char MJ1[21] = "   home position    ";

const unsigned char DATA[8] = {1, 3, 2, 6, 4, 12, 8, 9};

long int T;
int NS_Step, EW_Step, Z_Step;
int EW_Ref, NS_Ref, Z_Ref;
unsigned char RUN, SUN, MOON, STAR;


// Subroutine Declarations
#include <htc.h>

#include       "lcd_portd.h"
#include       "function.h"


// Subroutines
#include       "lcd_portd.c"
#include       "function.c"

// High-priority service
void interrupt IntServe(void){

  if (TMR2IF) {
  TRISD = 0;
```

```
    if (RUN) {
     T += 1;

     // Equations to use if SUN is pressed
     // Causes stepper motors to adjust at certain rate
                    if (SUN){
       EW_Ref = T * 0.0287005;                    // T * 1712.3 / 60000
       Z_Ref  = T * 0.000350185;                  // 12.7323 / 60000
       NS_Ref = (T * 0.000087078) * (T * 0.000087078);   // sqaureroot(.27297) / 120000
                    }

     // Equations to use if MOON is pressed
     // Causes stepper motors to adjust at certain rate
                   if (MOON){
       EW_Ref = T * 0.0275612;                    // T * 2273 / 60000
       Z_Ref  = T * 0.000336216;                  // 12.26767 / 60000
       NS_Ref = (T * 0.000086112) * (T * 0.000086112);    // squareroot(12.26767) / 120000
     }

     // Equations to use if STAR is pressed
     // Causes stepper motors to adjust at certain rate
            if (STAR){
              EW_Ref = T * 0.0287005;                    // T * 1722.03 / 60000
       Z_Ref  = T * 0.000350185;                  // 12.76423 / 60000
                     NS_Ref = (T * 0.000087078) * (T * 0.000087078);    // squareroot(.27297)/120000
    }
   }

    // Adjust stepper motors
    if (NS_Step < NS_Ref) NS_Step += 1;
    if (NS_Step > NS_Ref) NS_Step -= 1;
    if (EW_Step < EW_Ref) EW_Step += 1;
    if (EW_Step > EW_Ref) EW_Step -= 1;
    if (Z_Step < Z_Ref) Z_Step += 1;
    if (Z_Step > Z_Ref) Z_Step -= 1;

    // Make stepper motors run at specified rate
    PORTB = (DATA[Z_Step & 7] << 4) + DATA[EW_Step & 7];
    PORTA = DATA[NS_Step & 7];

    TMR2IF = 0;
  }
}

// Function to write steps taken on final display page
void LCD_Out(int DATA)
{
  unsigned char A[8], i;

  if (DATA < 0) {
    LCD_Write('-');
    DATA = -DATA;
  }

  else {
```

```
    LCD_Write(' ');
  }

  for (i=0; i<8; i++) {
    A[i] = DATA % 10;
    DATA = DATA / 10;
  }

  for (i=8; i>0; i--) {
    LCD_Write(A[i-1] + '0');
  }
}

// Function to display -time until +time
void Display_Time(void)
{
  int Min, Sec;
  unsigned char A[5], i;
  long int X;

  if (T < 0) {
    LCD_Write('-');
    X = -T;
  }

  else {
    LCD_Write(' ');
    X = T;
  }

  Min = X / 6000;
  Sec = X - (Min * 6000);
  A[1] = Min/10;
  A[0] = Min - A[1]*10;
  LCD_Write(A[1] + '0');
  LCD_Write(A[0] + '0');
  LCD_Write(':');

  for (i=0; i<4; i++) {
    A[i] = Sec % 10;
    Sec = Sec / 10;
  }

  LCD_Write(A[3] + '0');
  LCD_Write(A[2] + '0');
  LCD_Write('.');
  LCD_Write(A[1] + '0');
  LCD_Write(A[0] + '0');
}

int Linear(int X)
{
  float a;
  float Y;
```

```
  a = X;
  a = a / 3000.0;
  Y = a*T;
  return(Y);
}

int Quad(int X)
{
  float t;
  int Y;

  t = T / 3000.0;
  Y = t*t*X;
  return(Y);
}

// Function which moves the board to Start Position
void Goto_Start_Position(void)
{
  unsigned char i;

  LCD_Move(0,0);  for (i=0; i<20; i++) LCD_Write(ME0[i]);
  LCD_Move(1,0);  for (i=0; i<20; i++) LCD_Write(ME1[i]);
  LCD_Move(2,0);  for (i=0; i<20; i++) LCD_Write(MA3[i]);
  LCD_Move(3,0);  for (i=0; i<20; i++) LCD_Write(MA3[i]);

  T = -120000;
  RUN = 1;
  Wait_ms(100);

  while(EW_Step != EW_Ref) {
    T = -120000;
    }

  RUN = 0;

  LCD_Init();   // Needed or else Sun gets stuck in a loop when
            // motors are attached

  LCD_Move(0,0);  for (i=0; i<20; i++) LCD_Write(MF0[i]);
  LCD_Move(1,0);  for (i=0; i<20; i++) LCD_Write(MF1[i]);
  LCD_Move(2,0);  for (i=0; i<20; i++) LCD_Write(MF2[i]);
  LCD_Move(3,0);  for (i=0; i<20; i++) LCD_Write(MA3[i]);

}

// Function which moves the board to Home Position
void Goto_Home_Position(void)
{
  unsigned char i;

  LCD_Move(0,0);  for (i=0; i<20; i++) LCD_Write(MH0[i]);
  LCD_Move(1,0);  for (i=0; i<20; i++) LCD_Write(MH1[i]);
  LCD_Move(2,0);  for (i=0; i<20; i++) LCD_Write(MA3[i]);
  LCD_Move(3,0);  for (i=0; i<20; i++) LCD_Write(MA3[i]);
```

```
  T = 0;
  RUN = 1;
  Wait_ms(100);

  while(EW_Ref != EW_Step) {
    T = 0;
    }

  RUN = 0;
}

// Main Routine
void main(void)
{
  unsigned char i;
  unsigned int j;

  T1CON = 0;

  TRISA = 0;
  TRISB = 0;
  TRISC = 0;
  TRISD = 0;
  TRISE = 0;
  TRISA = 0;
  PORTB = 0;
  PORTC = 0;
  PORTD = 0;
  PORTE = 0;
  ADCON1 = 0x0F;

  LCD_Init();          // initialize the LCD

// Timer2 for 1ms

  T2CON   = 0xFF;      // 1ms interrupt rate
  PR2     = 197;       // 01001101
  TMR2IE  = 1;
  TMR2ON  = 1;
  TMR2IP = 1;


  T = -120000;
  j = 0;

  TRISC = 0xFF;

  EW_Step = 0;
  NS_Step = 0;
  Z_Step = 0;
  EW_Ref = 0;
  NS_Ref = 0;
  Z_Ref = 0;
  SUN=0;
```

```
  MOON=0;
  STAR=0;
  RUN = 0;

  GIE = 1;
  PEIE = 1;

// Adjust Z axis display screen
  LCD_Move(0,0);  for (i=0; i<20; i++) LCD_Write(MA0[i]);
  LCD_Move(1,0);  for (i=0; i<20; i++) LCD_Write(MA1[i]);
  LCD_Move(2,0);  for (i=0; i<20; i++) LCD_Write(MA2[i]);
  LCD_Move(3,0);  for (i=0; i<20; i++) LCD_Write(MA3[i]);

// Allow the user to manually adjust Z Rotation
 while(!RC4) {
    if (RC2) Z_Ref += 10;
    if (RC3) Z_Ref -= 10;
    Wait_ms(130);
    }
  Z_Step = 0;
  Z_Ref = 0;

// Proceed to E/W display screen after Enter is pressed
  while(RC4);
  Wait_ms(100);

// Level E/W axis display screen
  LCD_Move(0,0);  for (i=0; i<20; i++) LCD_Write(MB0[i]);
  LCD_Move(1,0);  for (i=0; i<20; i++) LCD_Write(MB1[i]);
  LCD_Move(2,0);  for (i=0; i<20; i++) LCD_Write(MB2[i]);
  LCD_Move(3,0);  for (i=0; i<20; i++) LCD_Write(MA3[i]);

// Allow the user to manually adjust E/W Axis
  while(!RC4) {
    if (RC2) EW_Ref += 10;
    if (RC3) EW_Ref -= 10;
    Wait_ms(130);
    }
  EW_Step = 0;
  EW_Ref = 0;

// Proceed to N/S display after Enter is pressed
  while(RC4);
  Wait_ms(100);

 // Level N/S axis display screen
  LCD_Move(0,0);  for (i=0; i<20; i++) LCD_Write(MC0[i]);
  LCD_Move(1,0);  for (i=0; i<20; i++) LCD_Write(MC1[i]);
  LCD_Move(2,0);  for (i=0; i<20; i++) LCD_Write(MC2[i]);
  LCD_Move(3,0);  for (i=0; i<20; i++) LCD_Write(MA3[i]);

// Allow the user to manually adjust N/S Axis
  while(!RC4) {
    if (RC2) NS_Ref += 10;
    if (RC3) NS_Ref -= 10;
```

```
    Wait_ms(130);
    }
  NS_Step = 0;
  NS_Ref = 0;

// Proceed to Home Position after Enter is pressed
  while(RC4);
  Wait_ms(100);

// Home Poistion
  LCD_Move(0,0);  for (i=0; i<20; i++) LCD_Write(MD0[i]);
  LCD_Move(1,0);  for (i=0; i<20; i++) LCD_Write(MA3[i]);
  LCD_Move(2,0);  for (i=0; i<20; i++) LCD_Write(MD1[i]);
  LCD_Move(3,0);  for (i=0; i<20; i++) LCD_Write(MD2[i]);

// Proceed to Start Position after MOON, STAR, or SUN button is pressed
  while(!RC1){

            if(RD0) {    //MOON button
                    MOON=1;
            Wait_ms(200);
                    Goto_Start_Position();
            RUN = 0;
            }

            if(RC5) {    //STAR button
                    STAR=1;
            Wait_ms(200);
                    Goto_Start_Position();
                    RUN = 0;
            }

            if(RD1) {   //SUN button
                    SUN=1;
            Wait_ms(200);
                    Goto_Start_Position();
                    RUN = 0;
            }
  }

  RUN = 0;

// Begin running the program after pressing Start
  while(RC1);

// Running display showing time and steps taken in all directions
  LCD_Move(0,0);  for (i=0; i<20; i++) LCD_Write(MG0[i]);
  LCD_Move(1,0);  for (i=0; i<20; i++) LCD_Write(MG1[i]);
  LCD_Move(2,0);  for (i=0; i<20; i++) LCD_Write(MG2[i]);
  LCD_Move(3,0);  for (i=0; i<20; i++) LCD_Write(MG3[i]);

// Start
  RUN = 1;

  while(1) {
```

```
    if (T > 120000) {

                    RUN = 0;

          LCD_Move(0,0);  for (i=0; i<20; i++) LCD_Write(MI0[i]);
          LCD_Move(1,0);  for (i=0; i<20; i++) LCD_Write(MA3[i]);
          LCD_Move(2,0);  for (i=0; i<20; i++) LCD_Write(MI1[i]);
          LCD_Move(3,0);  for (i=0; i<20; i++) LCD_Write(MI2[i]);

            Wait_ms(8000);

     LCD_Move(0,0);  for (i=0; i<20; i++) LCD_Write(MI0[i]);
          LCD_Move(1,0);  for (i=0; i<20; i++) LCD_Write(MA3[i]);
          LCD_Move(2,0);  for (i=0; i<20; i++) LCD_Write(MJ0[i]);
          LCD_Move(3,0);  for (i=0; i<20; i++) LCD_Write(MJ1[i]);

     Wait_ms(8000);

     if (RC0) Goto_Home_Position();
           }

           else {

          LCD_Move(0,10);  Display_Time();
          LCD_Move(1,10);  LCD_Out(EW_Ref);
          LCD_Move(2,10);  LCD_Out(NS_Ref);
          LCD_Move(3,10);  LCD_Out(Z_Ref);

           }
  }
}
```